Denis SINGER et Jean-Baptiste DARRAS
INSEE FRANCE

ATELIER outil d'aide au développement d'applications SAS/AF en version 6.08

F

Cet outil présenté au CLUB SAS 93 à Toulouse correspondait à des besoins dans le développement en version 6.06 et 6.07 de SAS.

La mise en production de la version 6.08 nous a conduit à redévelopper cet atelier pour la prise en compte de la Frame technologie et de la programmation orientée objet. Par ailleurs les possibilités de la Frame technologie nous ont permis de développer une fonction de MAINTENANCE qui n'existait pas dans la version antérieure du produit.

I. MISE EN OEUVRE

Les extensions de la version "CLUB SAS 93" n'existent actuellement que sur Windows.

Le travail avec souris, des temps de réponse corrects font qu'il est plus agréable de développer sur PC sous Windows et éventuellement de transporter le développement terminé sur MVS.

Une application AF développée en version 6.08 installe l'outil ATELIER sur le poste de travail du développeur et génère le lien avec une application AF existante.

Un dialogue très simple est donc nécessaire avec le développeur qui doit préciser :

- l'endroit (répertoire où il veut installer l'atelier,

- l'application qui fera appel à l'outil ATELIER.

Une fois l'application d'installation terminée le développeur possède une copie de l'ATELIER sous le répertoire qu'il a désigné, et le lien à l'outil est généré sous la forme d'un programme dans sa librairie applicative.

Pour ne pas relancer l'application d'installation sur chacune des applications qu'il veut développer ou maintenir, l'outil ATELIER met à la disposition du développeur une fonction nouvelle. Cette fonction nous l'avons appelée MAINTENANCE.

II - 1. MAINTENANCE

En résumé, et de façon pratique l'application d'Installation crée, dans l'espace applicatif du développeur, un programme.

Ce programme, appelé Premier.program possède trois fonctions matérialisées dans une "fonction Block" par les blocs.

a) Application

Exécution de l'application de l'utilisateur dont le point d'entrée a été mentionné dans le dialogue d'installation.

b) Développement

Exécution de l'application AF dite ATELIER qui va gérer l'application de l'utilisateur mentionnée dans le dialogue d'Installation ou celles qu'il aura indiqué dans la fonction Maintenance.

c) Maintenance

Permet à l'utilisateur de gérer le développement de plusieurs applications sans être obligé de repasser l'application d'Installation.

Voir Annexe 1

II - 2. MAINTENANCE - MODE OPERATOIRE

En cliquant sur le pavé maintenance vu plus haut deux actions sont proposées. Création/Exécution.

a) Création

L'utilisateur va créer sa propre Frame qui comportera autant d'ICON que d'application qu'il veut maintenir. Le dialogue qui s'en suit permet à l'utilisateur et pour chaque application de choisir:

- un modèle d'ICON,
- un nom d'application à l'ICON,
- le point d'entrée de l'application,

Point d'entrée qui est constitué par cliquage si la librairie est déjà allouée. Une étape supplémentaire est nécessaire pour allouer la librairie dans le cas contraire.

b) Exécution

La fonction Création a permis au développeur de créer une entrée de type Frame sans connaissance particulière et de générer automatiquement le lien à l'outil ATELIER.

En cliquant sur l'un des ICONS le choix sera donné à l'utilisateur d'exécuter son application: bloc APPLICATION, ou de la développer : bloc DEVELOPPEMENT.

Voir Annexes 2, 3, 4, 5

III. ATELIER DEVELOPPEMENT

L'ATELIER présenté au Club SAS 93 comportait trois écrans principaux :

- ATELIER DEVELOPPEMENT

- ATELIER GESTION

- ATELIER PROJET

F

L'extension liée à la version 6.08 ne remet pas en cause l'organisation des différents écrans de l'ATELIER. Des fonctions qui existaient ont été enrichies d'autres ont été créées introduisant des blocs de choix nouveaux dans des écrans existants.

Les objectifs : Confort de développement, cohérence entre l'objet actif ou exécutable et les différents objets qui ont participé à sa création ont été maintenus.

a) Préambule.

Comme les entrées de type PROGRAM étaient une nouveauté de la version 6.06, les entrées de type Frame sont une nouveauté de la version 6.08.

Pour créer une entrée de Type Frame, le produit SAS/AF place le développeur dans la fenêtre DISPLAY. Celui-ci en utilisant la souris va définir des régions, ensuite remplir ces régions par des classes d'objet. Ces classes d'objets sont stockées dans une entrée de type RESOURCE. Chaque classe possède un certain nombre de variables. Affecter une valeur à une variable de la classe en fait un objet. L'entrée de Type Frame est donc composée d'objets.

Le développeur peut utiliser les classes d'objets prédéfinies par SAS qui se trouvent dans l'entrée SASHELP.FSP.BUILD.RESOURCE. Il peut aussi vouloir définir ses propres classes et les rendre accessibles quand il voudra remplir une région.

L'outil ATELIER propose d'aider le développeur dans la création de ses classes d'objets par l'action **CLASS OBJET AF** et de retrouver ces classes par l'action **BUILD AF**

Ces deux choix font partie du premier écran de l'ATELIER - Ecran ATELIER DEVELOPPEMENT.

b) Normes

Les objets exécutables tels que les entrées de type program CBT, HELP, FRAME, SCL, PMENU etc sont dans le catalogue principal de l'application appelé aussi catalogue courant.

Il est souhaitable que le nom de ce catalogue soit le nom générique de l'application.

Les ressources utilisées pour construire les entrées FRAME sont dans l'entrée BUILD.RESOURCE de ce catalogue courant.

c) **CHOIX : BUILD** de l'écran ATELIER DEVELOPPEMENT

C'est l'exécution de la procédure BUILD AF sur le catalogue courant de l'application en ayant en ressource les "classes d'objets" qui se trouvent dans l'entrée BUID RESOURCE du catalogue courant.

SAS met à la disposition du développeur des classes d'objets qui se trouvent dans l'entrée SASHELP.FSP.BUILD.RESOURCE.

L'outil ATELIER met aussi à disposition des classes d'objets qui se trouvent dans l'entrée ATELIER.ATELIER.BUILD.RESOURCE.

Pour l'instant les classes développées sont peu nombreuses, elles servent essentiellement d'exemple.Mais on peut envisager des développements au sein de l'institut et des capitalisations grâce à l'ATELIER.

L'OUTIL ATELIER gère une application existante (tout au moins un répertoire existe, même vide). Deux cas de figure peuvent arriver.

c) - 1. BUILD RESOURCE EXISTE

L'Atelier va respecter les resources déjà existantes, l'utilisateur doit alors, s'il le désire intégrer manuellement les classes d'objets de l'ATELIER.

Une fonction **ATELIER** existe dans le choix **CLASS OBJET AF**. Cette fonction donne la documentation de ces classes et simule l'intégration dans les resssources de l'utilisateur.

c) - 2. BUILD RESOURCE N'EXISTE PAS

- Pas d'entrée BUILD.RESOURCE dans le catalogue courant de l'application.

Alors l'entrée BUILD.RESOURCE est créée automatiquement et elle contiendra les classes objets standard de SAS (SASHELP.FSP.BUILD.RESOURCE).

- Le développement n'est pas conforme à la norme ATELIER (catalogue courant pour l'entrée BUILD.RESOURCE) l'utilisateur doit mettre à jour ce catalogue en versant dans celui-ci les différentes CLASSES qu'il a pu déjà créer.

ATTENTION :

En exécution de Frames développées avec des ressources appartenant à des librairies différentes, le développeur devra allouer toutes ces librairies, puisque le nom de la ressource est stocké au niveau de l'entrée Frame.

Voir Annexe 6

d) CHOIX : CLASS OBJET AF.

Ce choix permet à l'utilisateur de l'ATELIER de développer ses propres classes d'objets et de les rendre disponibles pour développer ses entrées Frame.

d) - 1. Utilité de développer ses propres classes d'objets

A. Fréquence d'utilisation :

Le développeur d'entrées Frame peut faire appel fréquemment à un type d'objet. Par exemple une classe PUSH BUTTON qu'il instancie pour faire apparaître un texte dans le bouton : RETOUR et une action : commande CANCEL. Il est alors intéressant de développer une classe d'objet avec des attributs prédéfinis
(LABEL--> RETOUR, command processing --> CANCEL)



Le développeur n'aura plus besoin de définir ces attributs de la classe PUSH BUTTON et cela va lui assurer une certaine homogénéité dans ses développements.

B. Capitalisation :

Choisir une classe développée par quelqu'un d'autre.

Mettre les classes qu'il a développées à la disposition d'autres développeurs.

d) - 2. METHODOLOGIE

Pour développer une nouvelle classe d'objet le programmeur devra la définir à partir d'une classe parent. Il pourra garder ou modifier les attributs de la classe parent. pour définir ses propres attributs.

L'ATELIER ne va pas gérer l'ensemble des attributs.

Une partie de ceux-ci est affectée en interactif et le développeur va les retrouver facilement quand il sélectionnera sa classe d'Objet.

L'ATELIER va gérer les deux seuls attributs qui font partie des "additionnal attributes" qui sont le "SET CUSTOM ATTRIBUTES" et l'attribut "METHODS". Ceux-ci sont en fait les deux seuls associés à des entrées de catalogue.

" <u>CUSTOM ATTRIBUT</u> "

Pour une nouvelle classe, le développeur a le choix de garder le "CUSTOM ATTRIBUT" de la classe parent, soit définir ses propres attributs dans la fenêtre "SET CUSTOM ATTRIBUT"

Ces attributs personnalisées sont alors définis dans une entrée. L'ATELIER la définit comme une entrée de type Frame dans le catalogue ATTR.

C'est cette entrée qui sera exécutée quand l'utilisateur voudra remplir une zone d'une Frame avec la nouvelle classe créée.

" **METHOD** "

A chaque Classe est associée un ensemble de "methods".

Celles-ci vont définir les opérations que l'on pourra faire avec l'objet créé à partir de cette classe. Ces opérations sont invoquées par les routines CALL SEND et CALL NOTIFY dans le SCL.

Comme pour les attributs, la nouvelle classe hérite de l'ensemble des "METHODS" de sa classe parent.

Le développeur peut vouloir écrire ses propres "méthods" soit en modifiant une "Method" existante (override), soit en créant une nouvelle.

L'ATELIER les définit dans une entrée de type SCL dans le catalogue METHOD. Chaque label de cette entrée SCL est le nom d'une "METHOD".

" **RESOURCE** "

Le développeur ayant terminé d'écrire sa nouvelle classe d'objet en ayant saisi les bons noms d'entrées dans les fenêtres "SET CUSTOM ATTRIBUTES" et "METHODS". Il doit la rendre disponible pour le développement de ses nouvelles entrées Frame.

Pour ce faire il doit incorporer cette entrée de type "CLASS" dans l'entrée de type RESOURCE qui est le BUILD.RESOURCE du catalogue courant.

d) - 3. Architecture .

L'architecture proposée par l'outil ATELIER est basée sur des catalogues spécifiques.

- Catalogue COURANT :
    - contient les Frames, les program, les Help etc
    - l'entrée BUILD.RESOURCE

- Catalogue ATTR :
    - contient les entrées de type CLASS
    - Les "custom attributes" entrées de type Frame et SCL

- Catalogue METHOD :
    - contient les "methods" entrées de type SCL

- Catalogue DOC :
    - contient de la documentation des nouvelles classes d'objets.

A une classe d'objet nommée CLASS1 on aura donc :

- APPL.ATTR.CLASS1.CLASS

- APPL.ATTR.CLASS1.FRAME)
- APPL.ATTR.CLASS1.SCL) attribut

- APPL.METHOD.CLASS1.SCL "method"

- APPL.APPL.BUILD.RESOURCE "resource" (les classes disponibles
APPL étant le nom du catalogue courant de l'application.

- APPL.DOC.CLASS1.SOURCE La documentation

Voir Annexes 7 et 8

IV MODE OPERATOIRE

Pour développer ses propres classes d'objets.

- Au choix CLASS OBJET AF de l'écran ATELIER DEVELOPPEMENT est associé une liste de choix.

CREATION
SUPPRESSION
GESTION
ATELIER

IV - 1 CREATION

Ouverture de la fenêtre : IDENTIFICATION DE LA CLASSE D'OBJET

nom :
description :

Le champ description limité volontairement à une longueur de 12 caractères sera formatté avec le nom de l'application et l'dentifiant du développeur qui a conçu cette classe d'objet pour en faire une description complète de 39 caractères. Cette description sera reportée dans la description de classe et dans les descriptions des entrées associées ("CUSTOM ATTRIBUT", "METHOD").

La validation de cette fenêtre a pour effet d'enregistrer cette classe d'objet. Le développeur peut arrêter là ou poursuivre la création des entrées associées.

- Création du programme "ATTRIBUT"
- Création d'une "METHODE"
- Création de la "CLASS"
- Intégration dans les ressources
- Documentation

La création de ces différentes entrées se fait séquentiellement.

Mais ces entrées ne sont pas toutes nécessaires, aussi chaque fenêtre de création comporte les boutons < OK > < ANNUL > < Menu Principal >

- Menu Principal --> Arrêt de la création séquentielle

- ANNUL --> Arrêt de la création de l'entrée correspondante et on continue la création

Pour l'ATELIER, seule la première phase de création est obligatoire. Elle enregistre la nouvelle Classe d'Objet. Le développeur peut arrêter là et choisir GESTION pour développer un type d'entrée particulier.

IV - 2 GESTION

- Ouverture de la liste des classes d'objets déjà créées ou simplement enregistrées.

- Après avoir choisi une classe particulière le choix est donné au développeur.

CUSTOM ATTR
METHOD
CLASS
RESOURCE
DOCUMENTATION

pour créer une entrée associée à la classe qu'il a choisi.

Voir annexe 9

L'entrée de type source dans le catalogue DOC pour la classe d'objet est en partie renseignée par :

CLASS :
AUTEURS :
DATE :
DESCRIPTION :

IV - 3 SUPPRESSION

A pour effet de supprimer l'enregistrement de la classe d'objet. Les diverses entrées ayant pu être créées ne sont pas supprimées et la gestion des ressources est laissée au développeur.

IV - 4 ATELIER

C'est l'ouverture sur le monde extérieur. L'utilisateur peut avoir la documentation sur les classes disponibles avec l'atelier et le mode d'emploi pour se les récupérer.

Ce choix ATELIER va ouvrir une fenêtre qui ressemble fortement à la fenêtre de gestion des ressources.

Cette fenêtre sert à simuler les actions que l'utilisateur devra effectuer pour intégrer dans ses ressources une classe d'objet de l'ATELIER.

Elle comporte une liste des classes d'objet et un bouton GESTION. En cliquant sur une classe de la liste un choix est donné

ADD
DOC

Voir Annexes 10 et 11



DOC ---> affiche la fenêtre de type Source du catalogue DOC associé à la classe.

Voir Annexe 12

ADD ---> donne le chemin de recherche que l'utilisateur devra reproduire quand il aura cliqué su le bouton GESTION.

Le chemin formattera le nom de l'entrée CLASS c'est à dire :

Library
Catalog
Entry
Type

GESTION ---> Overture de la fenêtre BUILD.RESOURCE de l'utilisateur. Celui-ci n'a pu en cliquant qu'à reproduire les phases de ADD.

V NORME D'ORGANISATION DU DEVELOPPEMENT

Une application importante se fait souvent avec plusieurs développeurs. Chaque développeur travaille dans son espace, avec ses propres ressources. Une phase importante et sensible constitue à mettre en commun les différentes Frames créées par chacun. Il faut donc veiller à ce que chaque ressource soit disponible au moment de l'exécution du produit fini.

Trois méthodes sont possibles.

- Développer les classes d'objets au préalable et les copier dans chaque librairie de développement . Avec cette méthode on interdit aux développeurs de créer ses propres classes d'objets.

- Chaque développeur travaille dans sa propre librairie de développement . Mais toute l'équipe du projet a alloué un même nom logique à chaque librairie de développement et chaque développeur travaille dans un même nom de catalogue courant.

- Mettre en ligne toutes les librairies de développement avec le même nom logique qui a été donné au moment des développements.

LA NORME ATELIER.

- Le nom logique de chaque librairie de développement est le nom générique de l'Application (8caractères maximum).

- Le nom logique de la Librairie de production (produit fini) est aussi le nom de l'application.

- Le catalogue courant (ou principal) est aussi le nom générique de l'application. Ceci dans les librairies de développement et de production.

Les ressources auront alors un nom commun :

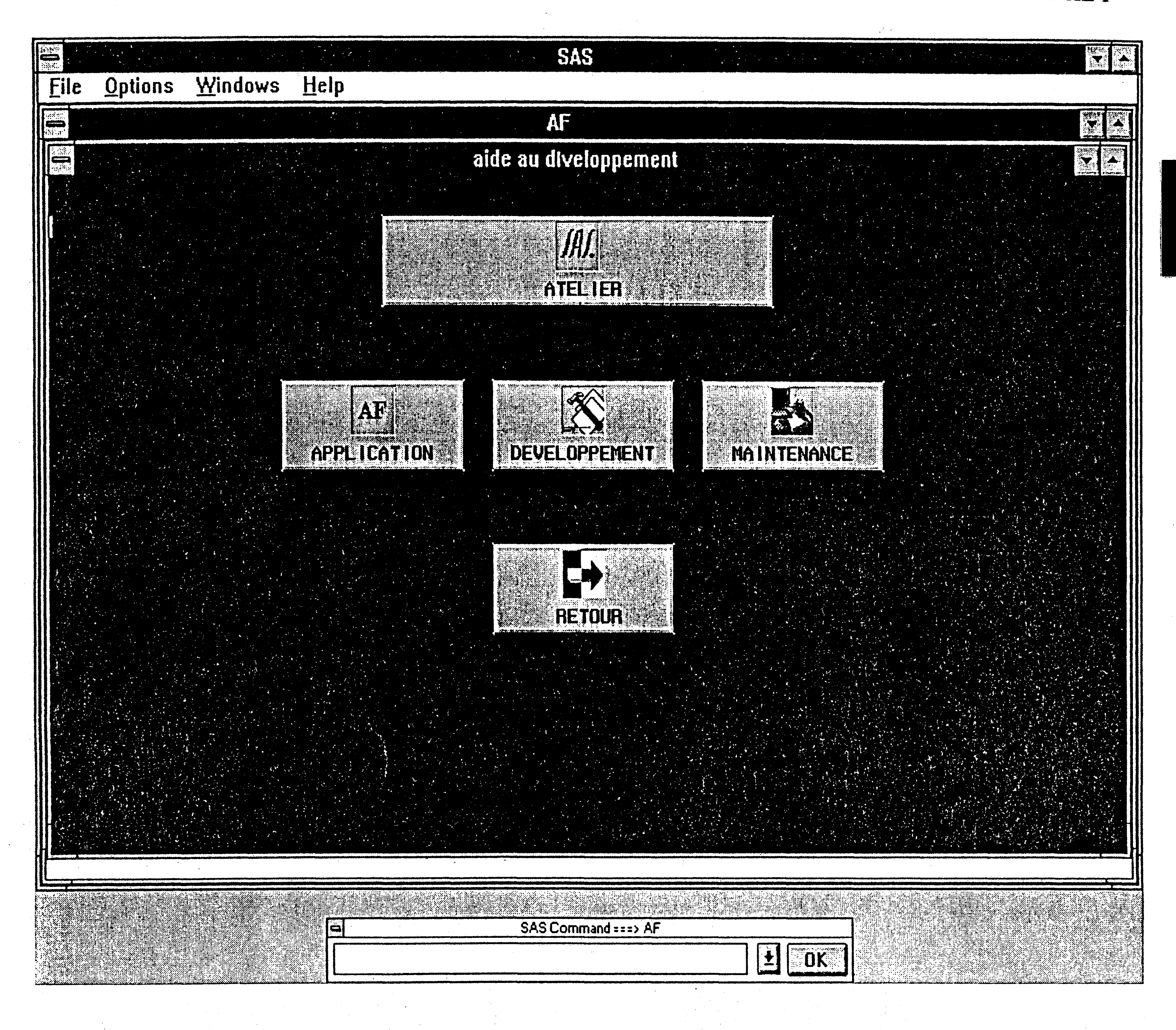
APPL.APPL.BUILD.RESOURCE (APPL étant le nom de l'application) et chaque frame retrouvera ainsi la ressource avec laquelle elle a été construite (une fois mis en commun les différentes ressources).

Exemple :

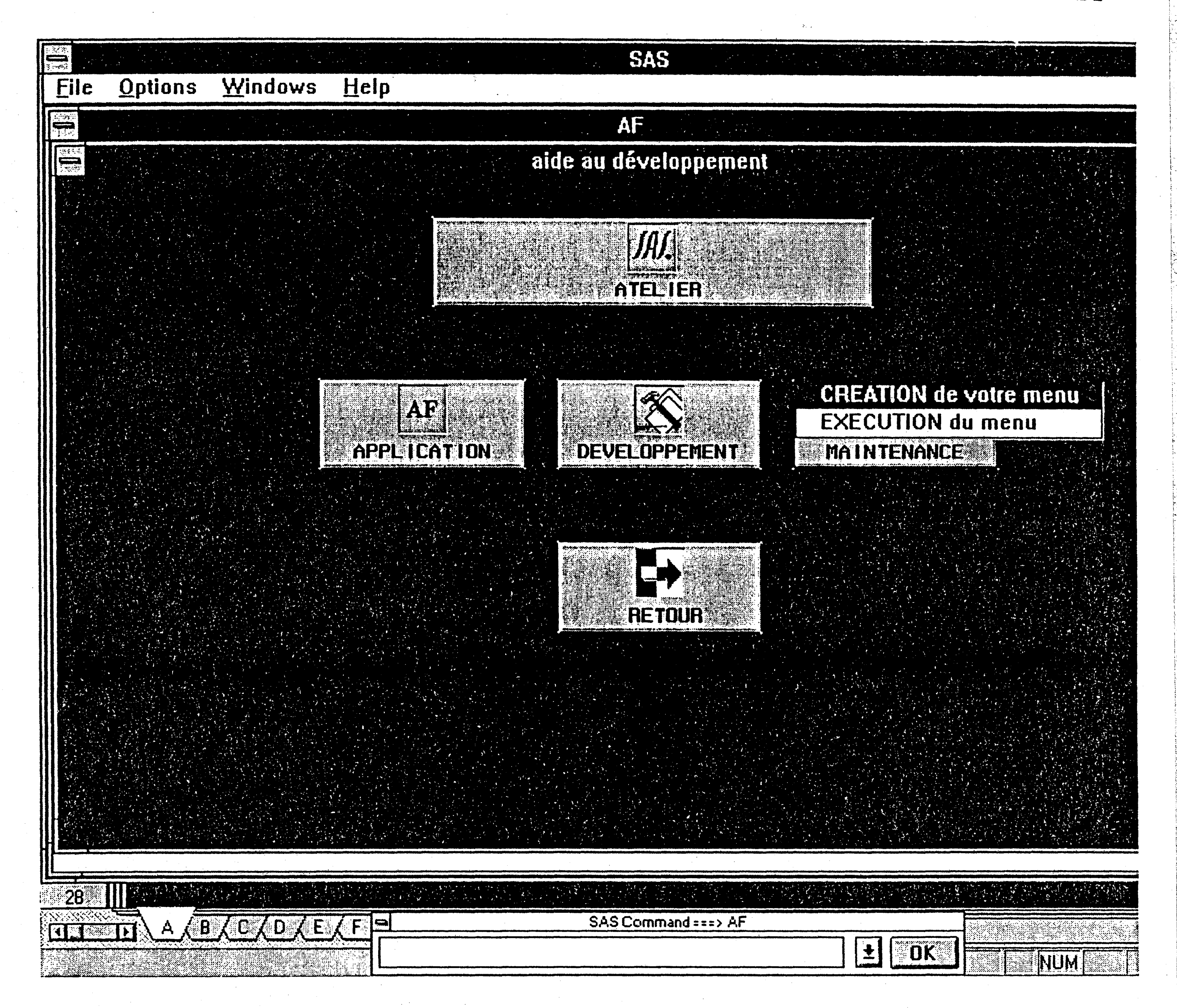
```
Développeur1 : Libname APPL "DSN1"
               proc build c = APPL.APPL;

Développeur2 : Libname APPL "DSN2"
               proc build c = APPL.APPL;

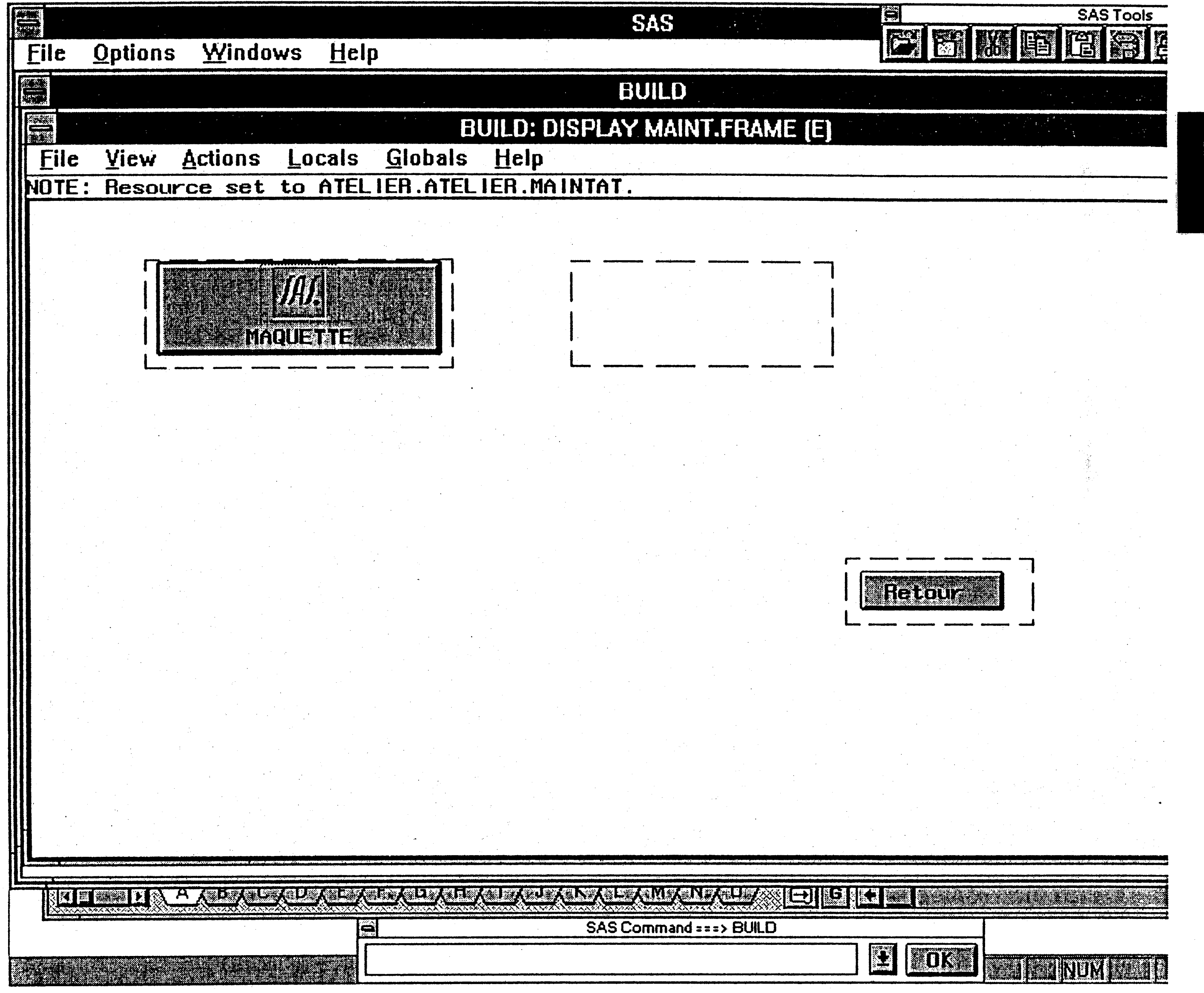
Production   : Libname APPL "DSN3";
```

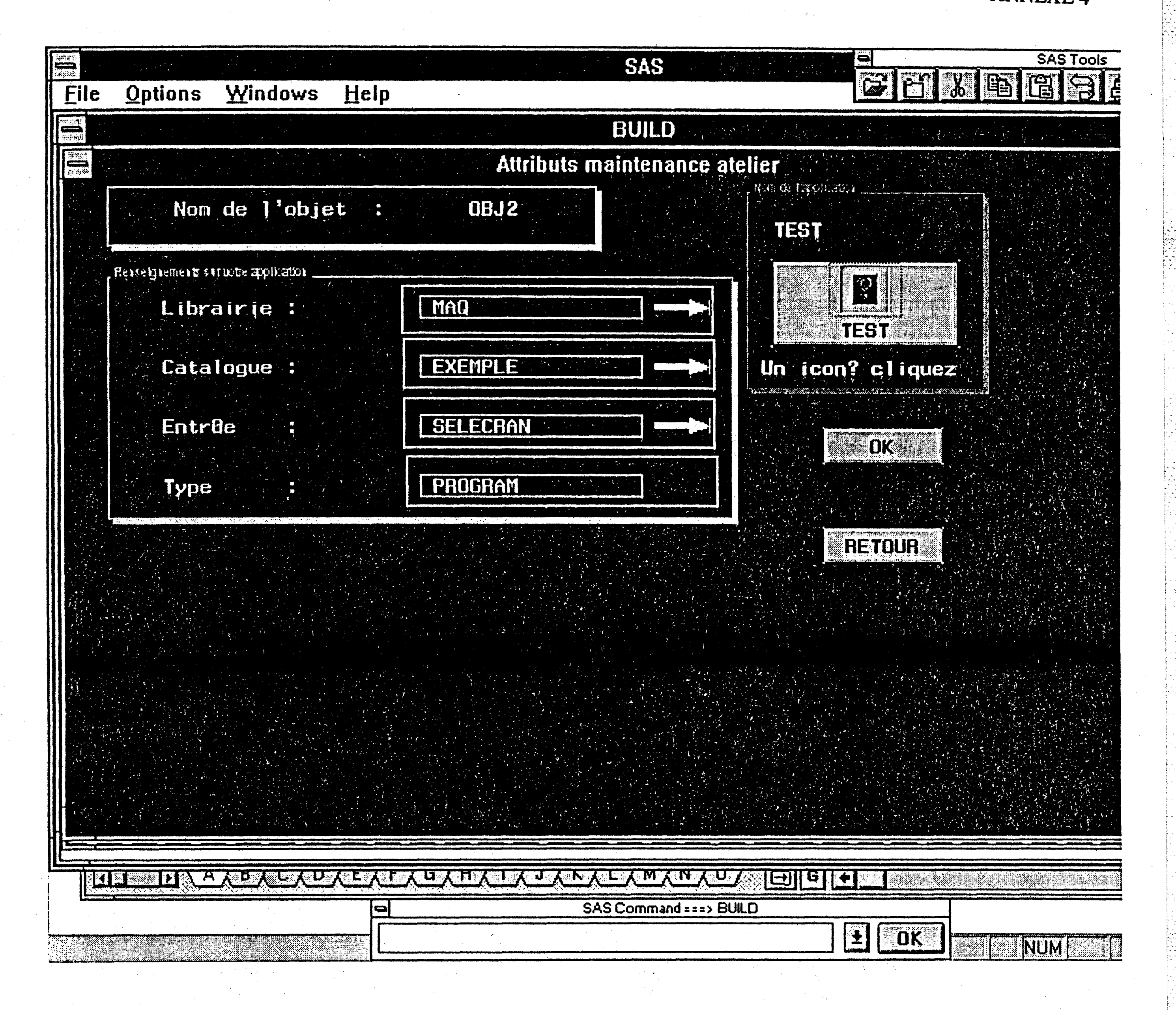

SAS
File Options Windows Help
AF
aide au dlveloppement
ATELIER
AF
APPLICATION
DEVELOPPEMENT
MAINTENANCE
RETOUR
SAS Command ===> AF
OK

F

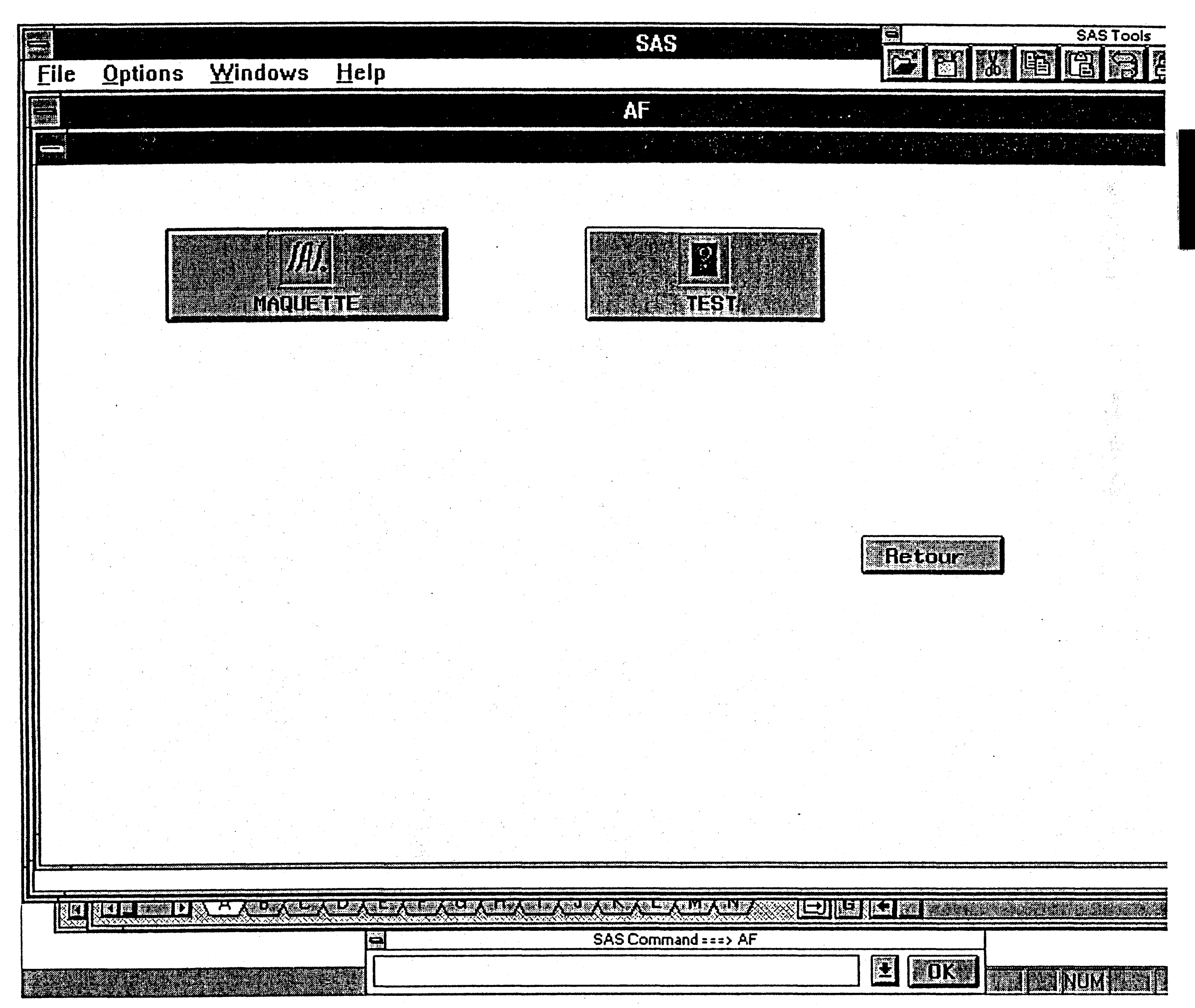

SAS
File Options Windows Help
AF
aide au développement
ATELIER
APPLICATION
DEVELOPPEMENT
CREATION de votre menu
EXECUTION du menu
MAINTENANCE
RETOUR
SAS Command ===> AF
OK
NUM

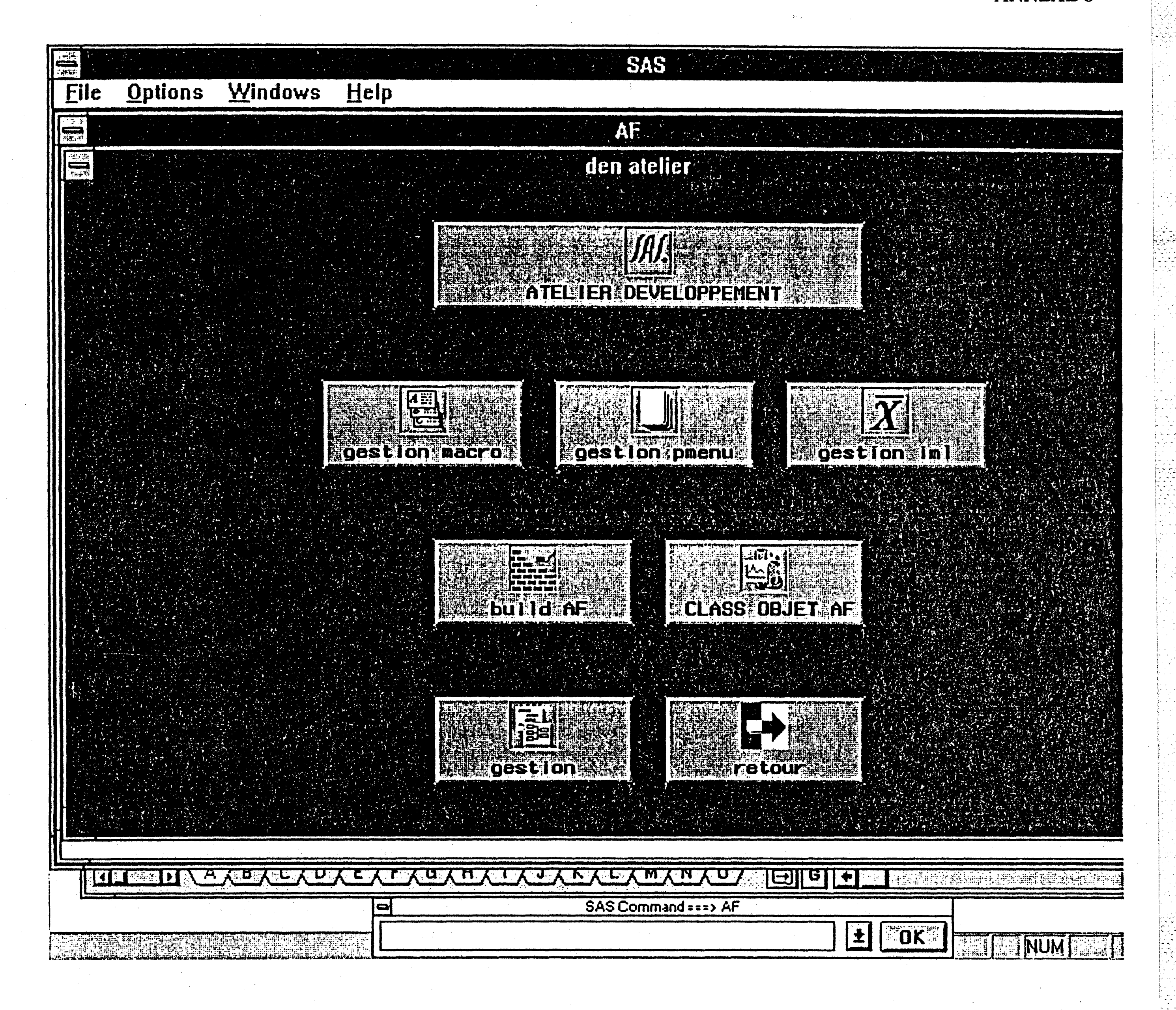
F


SAS
SAS Tools
File Options Windows Help
BUILD
BUILD: DISPLAY MAINT.FRAME (E)
File View Actions Locals Globals Help
NOTE: Resource set to ATELIER.ATELIER.MAINTAT.
MAQUETTE
Retour
SAS Command ===> BUILD
OK
NUM

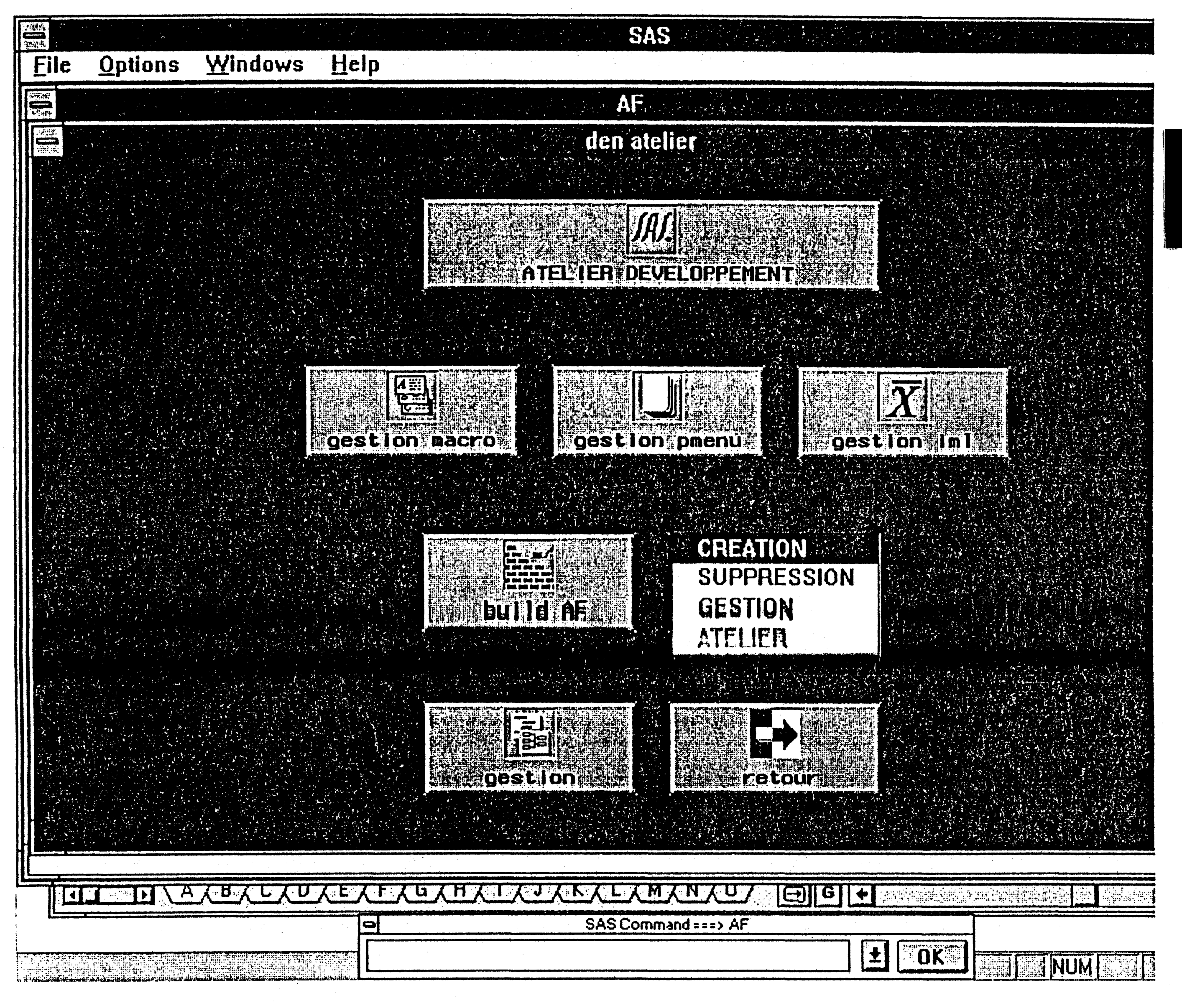

SAS
SAS Tools
File Options Windows Help
BUILD
Attributs maintenance atelier
Nom de l'objet : OBJ2
TEST
TEST
Un icon? cliquez
Librairie :
MAQ
Catalogue :
EXEMPLE
Entr0e :
SELECRAN
Type :
PROGRAM
OK
RETOUR
SAS Command ===> BUILD
OK
NUM

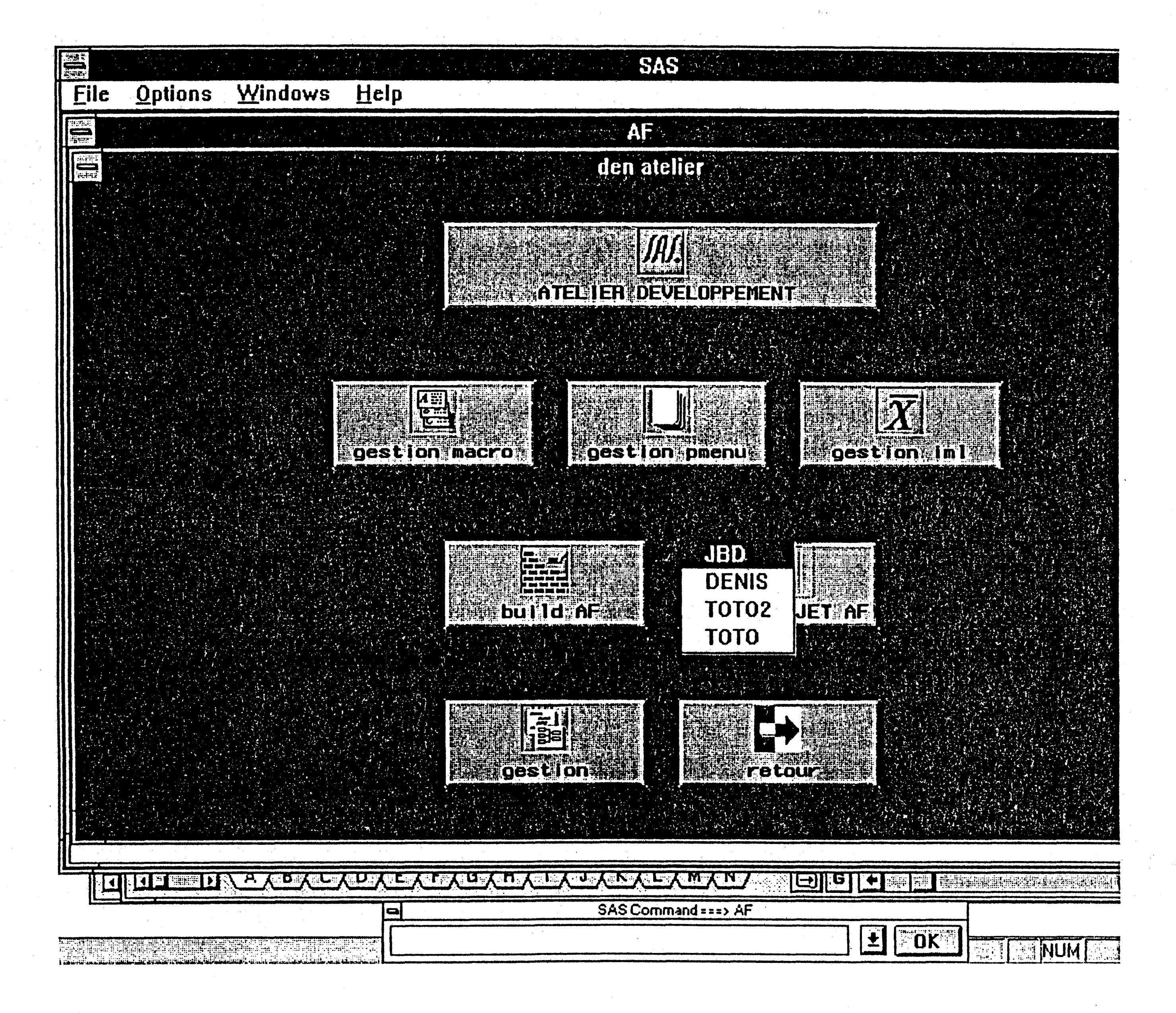
F


SAS
SAS Tools
File Options Windows Help
AF
MAQUETTE
TEST
Retour
SAS Command ===> AF
OK
NUM

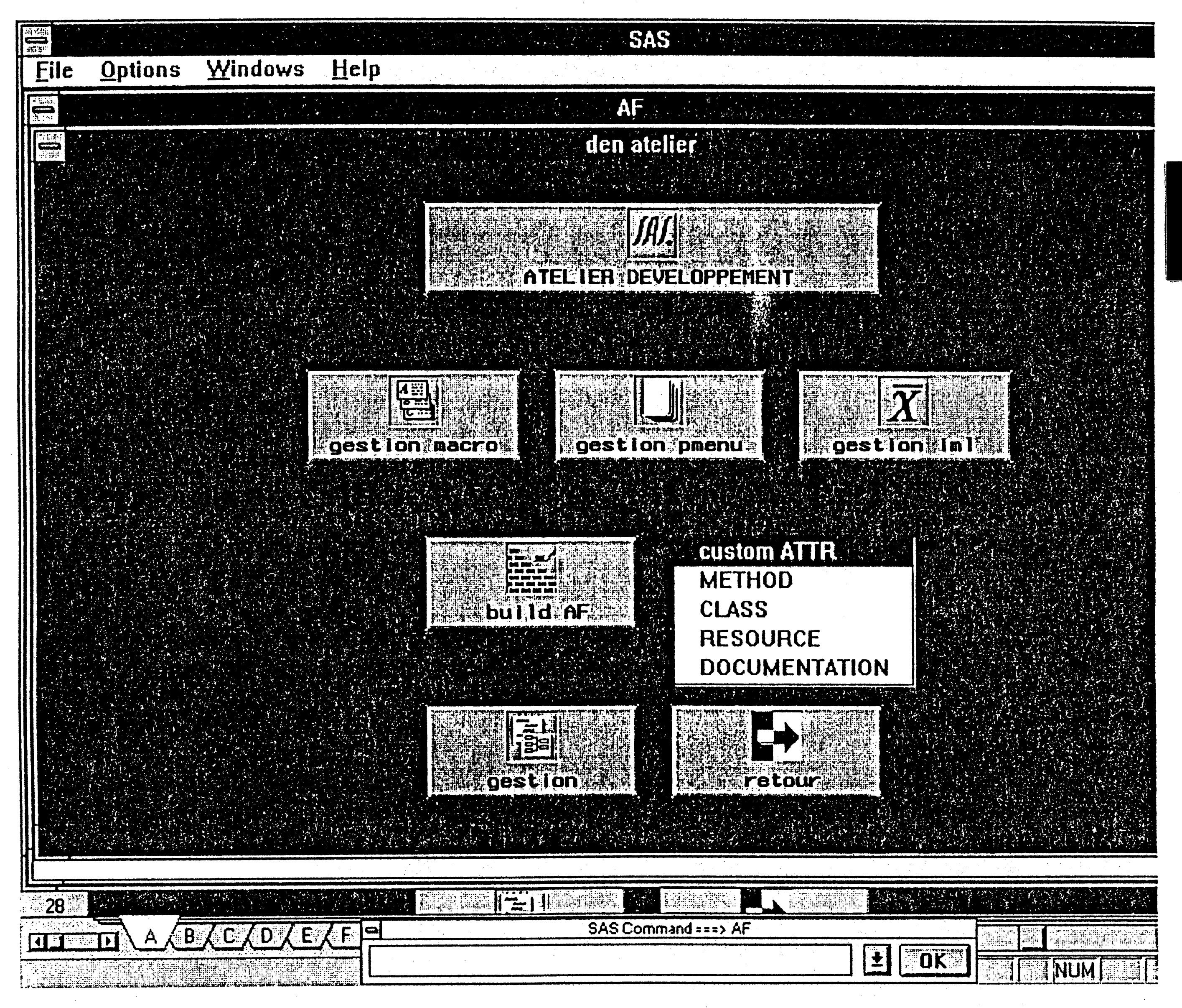

SAS
File Options Windows Help
AF
den atelier
ATELIER DEVELOPPEMENT
gestion macro
gestion pmenu
gestion iml
build AF
CLASS OBJET AF
gestion
retour
SAS Command ===> AF
OK
NUM

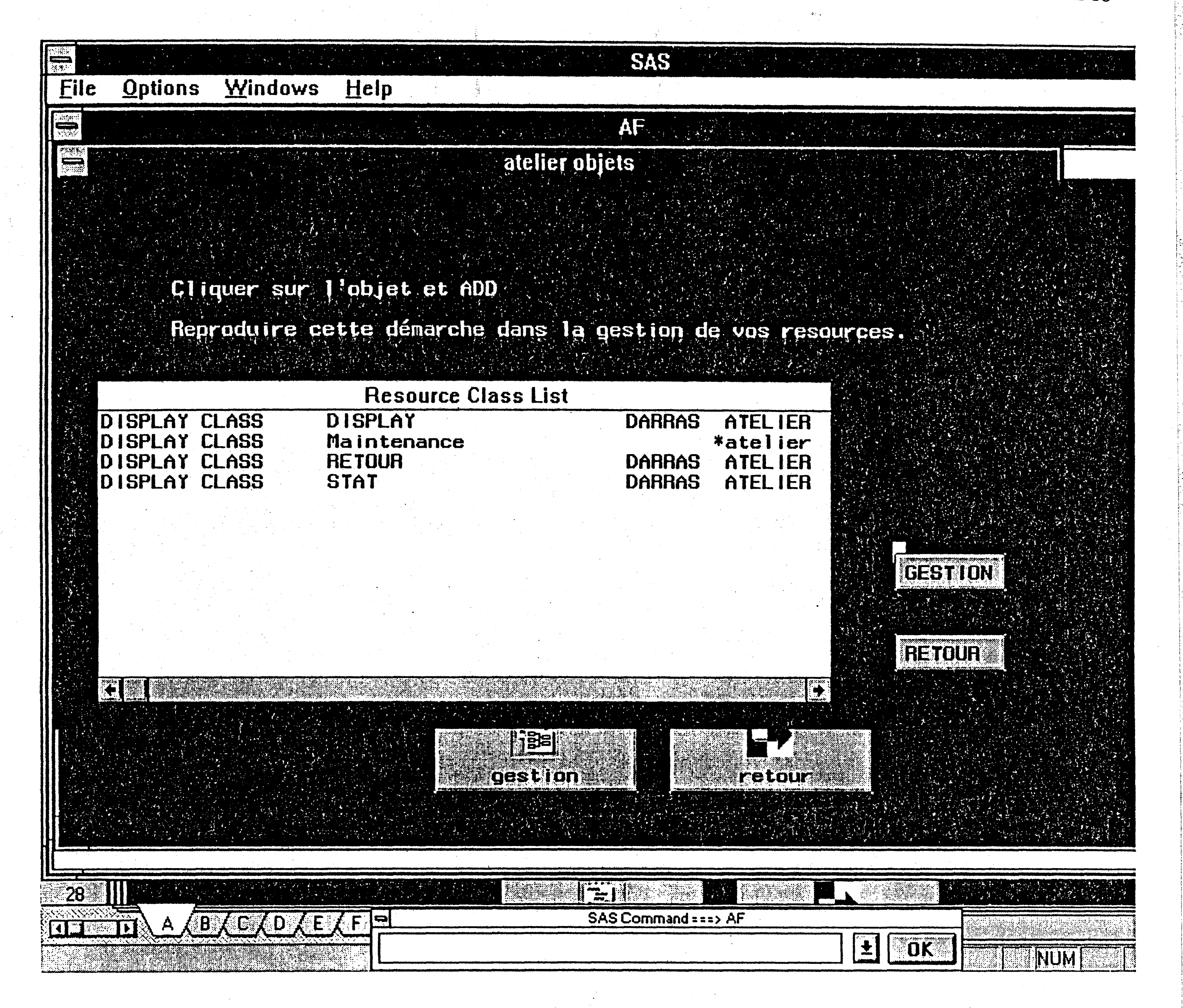
F

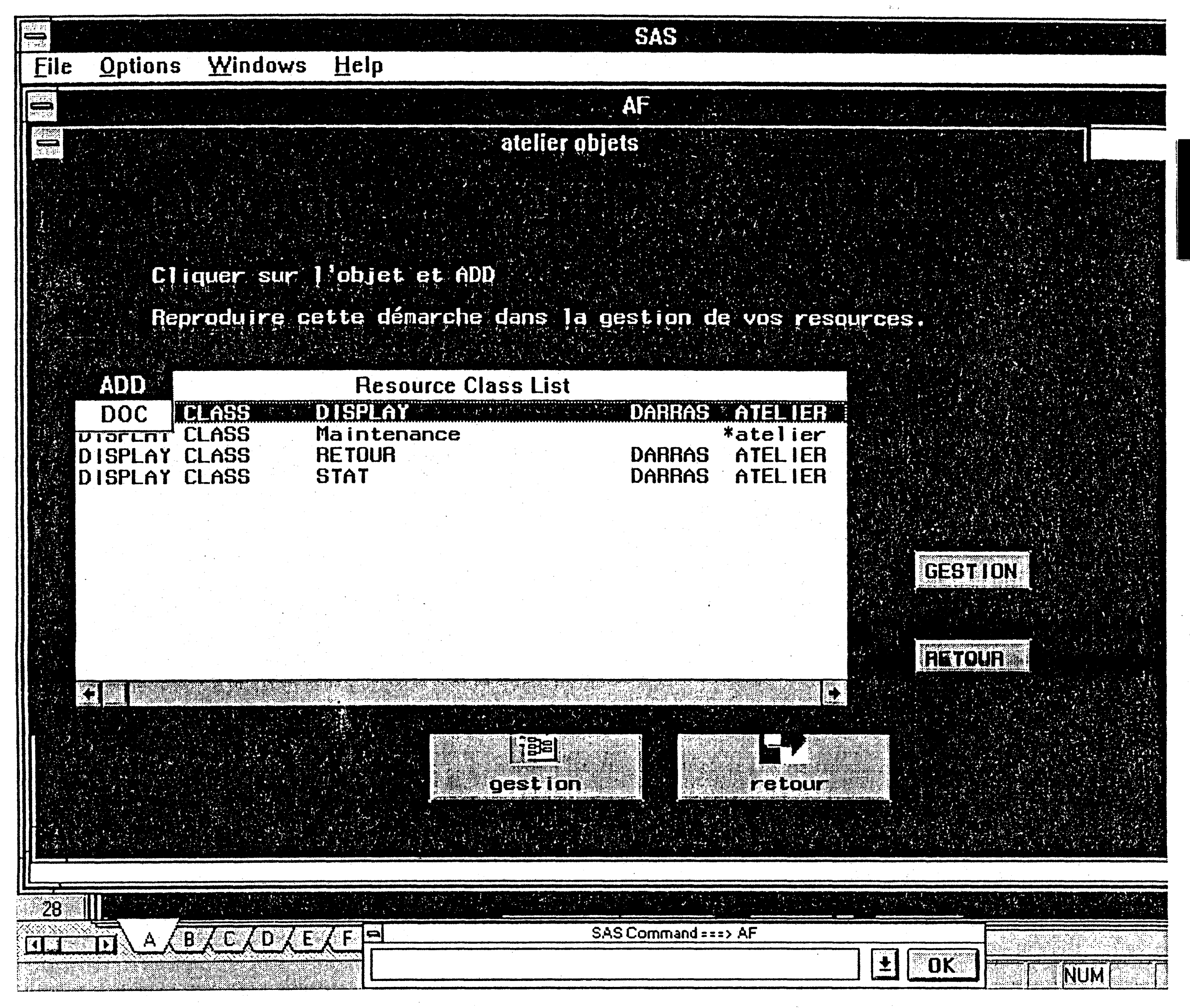

SAS
File Options Windows Help
AF
den atelier
ATELIER DEVELOPPEMENT
gestion macro
gestion pmenu
gestion iml
build AF
CREATION
SUPPRESSION
GESTION
ATELIER
gestion
retour
A B C D E F G H I J K L M N O
SAS Command ===> AF
OK
NUM


SAS
File Options Windows Help
AF
den atelier
ATELIER DEVELOPPEMENT
gestion macro
gestion pmenu
gestion iml
build AF
JBD
DENIS
TOTO2
TOTO
JET AF
gestion
retour
SAS Command ===> AF
OK
NUM

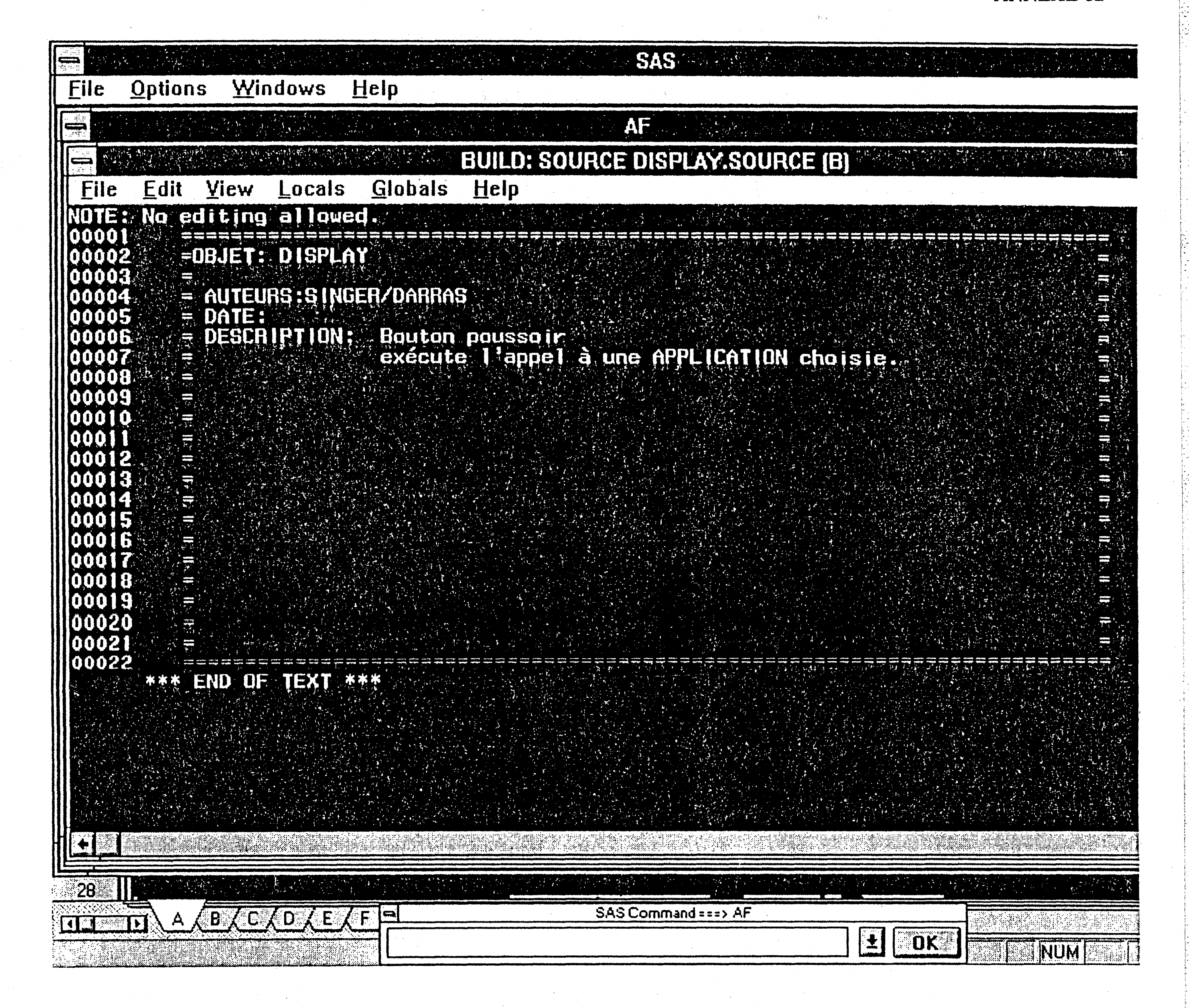
F


SAS
File Options Windows Help
AF
den atelier
ATELIER DEVELOPPEMENT
gestion macro
gestion pmenu
gestion iml
build AF
custom ATTR
METHOD
CLASS
RESOURCE
DOCUMENTATION
gestion
retour
28
A
B
C
D
E
F
SAS Command ===> AF
OK
NUM


SAS
File Options Windows Help
AF
atelier objets
Cliquer sur l'objet et ADD
Reproduire cette démarche dans la gestion de vos resources.
Resource Class List
DISPLAY CLASS DISPLAY DARRAS ATELIER
DISPLAY CLASS Maintenance *atelier
DISPLAY CLASS RETOUR DARRAS ATELIER
DISPLAY CLASS STAT DARRAS ATELIER
GESTION
RETOUR
gestion
retour
28
A B C D E F
SAS Command ===> AF
OK
NUM


SAS
File Options Windows Help
AF
atelier objets
Cliquer sur l'objet et ADD
Reproduire cette démarche dans la gestion de vos resources.
ADD
DOC
Resource Class List
CLASS DISPLAY DARRAS ATELIER
CLASS Maintenance *atelier
DISPLAY CLASS RETOUR DARRAS ATELIER
DISPLAY CLASS STAT DARRAS ATELIER
GESTION
RETOUR
gestion
retour
28
A B C D E F
SAS Command ===> AF
OK
NUM

F


SAS
File Options Windows Help
AF
BUILD: SOURCE DISPLAY.SOURCE (B)
File Edit View Locals Globals Help
NOTE: No editing allowed.
00001 ===
00002 =OBJET: DISPLAY =
00003 = =
00004 = AUTEURS:SINGER/DARRAS =
00005 = DATE: =
00006 = DESCRIPTION: Bouton poussoir =
00007 = exécute l'appel à une APPLICATION choisie. =
00008 = =
00009 = =
00010 = =
00011 = =
00012 = =
00013 = =
00014 = =
00015 = =
00016 = =
00017 = =
00018 = =
00019 = =
00020 = =
00021 = =
00022 ===
*** END OF TEXT ***
28
A B C D E F
SAS Command ===> AF
OK
NUM